AALBORG UNIVERSITY

Classification using Hierarchical Naïve Bayes models

by

Helge Langseth and Thomas D. Nielsen

TR-02-004 October 2002

Department of Computer Science
Decision Support Systems Group
Fredrik Bajers Vej 7E — DK 9220 Aalborg — Denmark

Classification using Hierarchical Naïve Bayes models

Helge Langseth
Dept. of Mathematical Sciences
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
helgel@math.ntnu.no

Thomas D. Nielsen
Dept. of Computer Science
Aalborg University
DK-9220 Aalborg Øst, Denmark
tdn@cs.auc.dk

Abstract

Classification problems have a long history in the machine learning literature. One of the simplest, and yet most consistently well performing set of classifiers is the Naïve Bayes models. However, an inherent problem with these classifiers is the assumption that all attributes used to describe an instance are conditionally independent given the class of that instance. When this assumption is violated (which is often the case in practice) it can reduce classification accuracy due to "information double-counting" and interaction omission.

In this paper we focus on a relatively new set of models, termed Hierarchical Naïve Bayes models. Hierarchical Naïve Bayes models extend the modelling flexibility of Naïve Bayes models by introducing latent variables to relax some of the independence statements in these models. We propose a simple algorithm for learning Hierarchical Naïve Bayes models in the context of classification. Experimental results show that the learned models can significantly improve classification accuracy as compared to other frameworks. Furthermore, the algorithm gives an explicit semantics for the latent structures (both variables and states), which enables the user to reason about the classification of future instances and thereby boost the user's confidence in the model used.

1 Introduction

Classification is the task of predicting the class of an instance from a set of attributes describing that instance, i.e., to apply a mapping from the attribute space into a predefined set of classes. When learning a classifier we seek to generate such a mapping based on a database of labelled instances. Classifier learning, which has been an active research field over the last decades, can therefore be seen as a model selection process where the task is to find the single model, from some set of models, with the highest classification accuracy. The *Naïve Bayes* (NB) models (Duda and Hart 1973) is a set of particularly simple models which has shown to offer very good classification accuracy. NB models assume that all attributes are conditionally independent given the class, but this assumption is clearly violated in many real world problems; in such situations overlapping information is counted twice by the classifier. To resolve this problem, methods for handling the conditional dependence between the attributes have become a lively research area; these methods are typically grouped into three categories: *Feature selection* (Kohavi and John 1997), *feature grouping* (Kononenko 1991; Pazzani 1995), and *correlation modelling* (Friedman et al. 1997).

The approach taken in this paper is based on correlation modelling using Hierarchical Naïve Bayes (HNB) models, see (Zhang et al. 2002). HNBs are tree-shaped Bayesian networks, with latent variables between the class node (the root of the tree) and the attributes (the leaves), see Figure 1. The latent variables are introduced to relax some of the independence statements of the NB classifier. For example, in the HNB model shown in Figure 1, the attributes A_1 and A_2 are not independent given C because the latent variable L_1 is unobserved. Note that if there are no latent variables in the HNB, it reduces to an NB model.

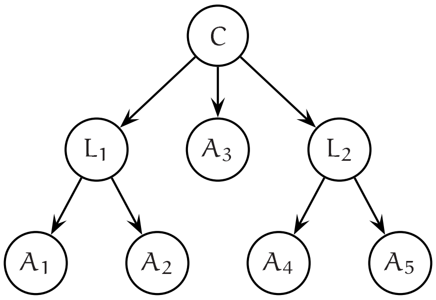


Figure 1: An HNB designed for classification. The class attribute C is in the root, and the attributes $\mathcal{A} = \{A_1, \ldots, A_5\}$ are leaf nodes. L_1 and L_2 are latent variables.

The idea to use HNBs in classification was first explored by Zhang et al. (2002). Zhang et al. (2002) search for the model maximizing the BIC score, which is a form of penalized log likelihood, see (Schwarz 1978); hence they look for a *scientific model* (Cowell et al. 1999) where the key is to find an interesting latent structure. In this paper we take the *technological modelling approach*: Our goal is mainly to build an accurate classifier. As a spin-off we also provide the latent variables with an explicit semantics, including a semantics for the state-spaces: Informally, a latent variable can be seen as aggregating the information from its children which is relevant for classification. Such a semantic interpretation is extremely valuable for a decision maker employing a classification system, as she can inspect the classification model and extract the "rules" which the system uses for the classification task.

The remainder of this paper is organized as follows: In Section 2 we give a brief overview of some approaches to Bayesian classification, followed by an introduction to HNB models in Section 3. In Section 4 we present an algorithm for learning HNB classifiers form data, and Section 5 is devoted to empirical results. We discuss some aspects of the algorithm in further detail in Section 6 and conclude in Section 7.

2 Bayesian classifiers

A Bayesian network (BN) (Pearl 1988; Jensen 2001) is a powerful tool for knowledge representation, as it provides a compact representation of a joint probability distribution over a set of variables. Formally, a BN over a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_m\}$ is denoted by $B = (B_S, \mathbf{\Theta}_{B_S})$, where B_S is a directed acyclic graph and $\mathbf{\Theta}_{B_S}$ is the set of conditional probabilities. To describe B_S, we let $\mathrm{pa}(X_i)$ denote the parents of X_i in B_S, we use $\mathrm{sp}(X_i)$ to denote the state-space of X_i, and for a set of variables we have $\mathrm{sp}(\mathcal{X}) = \times_{X \in \mathcal{X}} \mathrm{sp}(X)$. In the context of classification, we shall use C to denote the class variable ($\mathrm{sp}(C)$ is the set of possible classes), and $\mathcal{A} = \{A_1, \ldots, A_n\}$ is the set of attributes describing the possible instances to be classified.

When doing classification in a probabilistic framework, a new instance (described by $\mathbf{a} \in \mathrm{sp}(\mathcal{A})$) is classified to class c^* according to:

$$c^* = \arg \min_{c \in \mathrm{sp}(C)} \sum_{c' \in \mathrm{sp}(C)} L(c, c') P(C = c' \mid \mathbf{a}),$$

where $L(\cdot, \cdot)$ defines the *loss function*, i.e., $L(c, c')$ is the cost of classifying an instance to class c when the correct class is c'. The two most commonly used loss functions are the $0/1$-loss and the log-loss: The $0/1$-loss is defined s.t. $L(c, c') = 0$ if $c' = c$ and 1 otherwise, and the log-loss is given by $L(c, c') = \log(P(c' \mid \mathbf{a}))$ independently of c.

Since we rarely have access to $P(C = c \mid \mathbf{A})$, learning a classifier amounts to estimating this probability distribution from a set of labelled training samples which we denote by $\mathcal{D}_N = \{\mathbf{D}_1, \ldots, \mathbf{D}_N\}$; N is the number of training instances and $\mathbf{D}_i = \left(c^{(i)}, a_1^{(i)}, \ldots, a_n^{(i)}\right)$ is the class and attributes of instance i, $i = 1, \ldots, N$. Let $P(C = c \mid \mathbf{A}, \mathcal{D}_N)$ be the *a posteriori* conditional probability for $C = c$ given $\mathbf{A}$ after observing $\mathcal{D}_N$. Then an optimal Bayes classifier will classify a new instance with attributes $\mathbf{a}$ to class c^* according to (see e.g. (Mitchell 1997)):

$$c^* = \arg\min_{c \in \mathrm{sp}(C)} \sum_{c' \in \mathrm{sp}(C)} L(c, c') P(C = c' \mid \mathbf{a}, \mathcal{D}_N). \tag{2}$$

An immediate approach to estimate $P(C = c \mid \mathbf{A})$ is to use a standard BN learning algorithm, where the training data is used to give each possible classifier a *score* which signals its appropriateness as a classification model. One such scoring function is based on the minimum description length (MDL) principle (Rissanen 1978; Lam and Bacchus 1994):

$$\mathrm{MDL}(B \mid \mathcal{D}_N) = \frac{\log N}{2} \left| \widehat{\mathbf{\Theta}}_{B_S} \right| - \sum_{i=1}^{N} \log\left(P_B\left(c^{(i)}, \mathbf{a}^{(i)} \,\middle|\, \widehat{\mathbf{\Theta}}_{B_S}\right)\right). \tag{3}$$

That is, the best scoring model is the one that minimizes $\mathrm{MDL}(\cdot \mid \mathcal{D}_N)$, where $\widehat{\mathbf{\Theta}}_{B_S}$ is the maximum likelihood estimate of the parameters in the model, and $\left|\widehat{\mathbf{\Theta}}_{B_S}\right|$ is the dimension of the parameter space (i.e., the number of free parameters in the model). However, as pointed out in (Greiner et al. 1997; Friedman et al. 1997) a "global" criteria like MDL may not be well suited for learning a classifier, as:

$$\sum_{i=1}^{N} \log\left(P_B\left(c^{(i)}, \mathbf{a}^{(i)}\right)\right) = \sum_{i=1}^{N} \log\left(P_B\left(c^{(i)} \,\middle|\, \mathbf{a}^{(i)}\right)\right) + \sum_{i=1}^{N} \log\left(P_B\left(a_1^{(i)}, \ldots, a_n^{(i)}\right)\right).$$

In the equation above, the first term on the right-hand side measures how well the classifier performs on $\mathcal{D}_N$, whereas the second term measures how well the classifier estimates the joint distribution over the attributes. Thus, only the first term is related to the classification task, and the latter term will therefore merely bias the model search; in fact, the latter term will dominate the score if n is large. To overcome this problem, Friedman et al. (1997) propose to replace MDL with *predictive* MDL, MDL_p, defined as:

$$\mathrm{MDL}_p(B \mid \mathcal{D}_N) = \frac{\log N}{2} \left| \widehat{\mathbf{\Theta}}_{B_S} \right| - \sum_{i=1}^{N} \log\left(P_B\left(c^{(i)} \,\middle|\, \mathbf{a}^{(i)}, \widehat{\mathbf{\Theta}}_{B_S}\right)\right). \tag{4}$$

However, as also noted by Friedman et al. (1997), $\sum_{i=1}^{N} \log\left(P_B\left(c^{(i)} \,\middle|\, \mathbf{a}^{(i)}, \widehat{\mathbf{\Theta}}_{B_S}\right)\right)$ cannot be calculated efficiently in general.

The argument leading to the use of predictive MDL as a scoring function rests upon the asymptotic theory of statistics. That is, model search based on MDL_p is guaranteed to select the best classifier w.r.t. both log-loss and 0/1-loss when $N \to \infty$. Unfortunately, though, the score may not be successful for finite data sets (Friedman 1997). To overcome this potential drawback, Kohavi and John (1997) describe the *wrapper approach*. Informally, this method amounts to estimating the accuracy of a given classifier by cross validation (based on the *training data*), and to use this estimate as the scoring function. The wrapper approach relieves the scoring function from being based on approximations of the classifier design, but at the potential cost of higher computational complexity. In order to reduce this complexity when learning a classifier, one approach is to focus on a particular sub-class of BNs. Usually, these sub-classes are defined by the set of independence statements they encode. For instance, one such restricted set of BNs is the Naïve Bayes models which assume that $P(C|\mathbf{A}) \propto P(C) \prod_{i=1}^{n} P(A_i|C)$, i.e., that $A_i \perp\!\!\!\perp A_j \mid C$.

Even though the independence statements of the NB models are often violated in practice, these models have shown to provide surprisingly good classification results. Resent research into explaining the merits of the NB model has emphasized the difference between the 0/1-loss function and the log-loss, see e.g. (Friedman 1997; Domingos and Pazzani 1997). Friedman (1997, p. 76) concludes:

> Good probability estimates are not necessary for good classification; similarly, low classification error does not imply that the corresponding class probabilities are being estimated (even remotely) accurately.

The starting point of Friedman (1997) is that a classifier learned for a particular domain is a function of the training set. As the training set is considered a random sample from the domain, the classifier generated by a learner can be seen as a random variable; we shall use $\hat{P}(C = c \mid \mathbf{A})$ to denote the learned classifier. Friedman (1997) characterizes a classifier based on its bias (i.e., $\mathbb{E}_{\mathcal{D}_N}\left[P(C \mid \mathbf{A}) - \hat{P}(C \mid \mathbf{A})\right]^2$) and its variance (i.e., $\mathrm{Var}_{\mathcal{D}_N}\left(\hat{P}(C \mid \mathbf{A})\right)$); the expectations are taken over all possible training sets of size N. Friedman (1997) shows that in order to learn classifiers with low 0/1-loss it may not be sufficient to simply focus on finding a model with low classifier bias; robustness in terms of low classifier variance can be just as important.

An example of a class of models where low bias (i.e., fairly high model expressibility) is combined with robustness is the *Tree Augmented Naïve Bayes* (TAN) models, see (Friedman et al. 1997). TAN models relax the NB assumption by allowing a more general correlation structure between the attributes. More specifically, a Bayesian network model is initially created over the variables in $\mathcal{A}$, and this model is designed s.t. each variable A_i has at most one parent (that is, the structure is a *directed tree*). Afterwards, the class attribute is included in the model by making it the *parent* of each attribute. Friedman et al. (1997) use an adapted version of the algorithm by Chow and Liu (1968) to learn the classifier, and they prove that the structure they find is the TAN which maximizes the likelihood of $\mathcal{D}_N$; the algorithm has time complexity $O\left(n^2(N + \log(n))\right)$.

3 Hierarchical Naïve Bayes models

A special class of Bayesian networks is the so-called Hierarchical Naïve Bayes (HNB) models, a concept first introduced by Zhang et al. (2002), see also (Zhang 2002; Kočka and Zhang 2002). An HNB is a tree-shaped Bayesian network, where the variables are partitioned into three disjoint sets: $\{C\}$ is the class variable, $\mathcal{A}$ is the set of attributes, and $\mathcal{L}$ is a set of *latent* (or *hidden*) variables. In the following we use A to represent an attribute, whereas L is used to denote a latent variable; X and Y denote variables that may be either attributes or latent variables. In an HNB the class variable C is the root of the tree ($\mathrm{pa}(C) = \emptyset$) and the attributes are at the leaves ($\mathrm{ch}(A) = \emptyset, \forall A \in \mathcal{A}$); the latent variables are all internal ($\mathrm{ch}(L) \neq \emptyset$, $\mathrm{pa}(L) \neq \emptyset$, $\forall L \in \mathcal{L}$). The use of latent variables allows conditional dependencies to be encoded in the model (as compared to e.g. the NB model). For instance, by introducing a latent variable as a parent of the attributes A_i and A_j, we can represent the (local) dependence statement $A_i \not\perp\!\!\!\perp A_j \mid C$. Being able to model such local dependencies is particularly important for classification, as overlapping information would otherwise be double-counted. Note that the HNB model reduces to the NB model in the special case when there are no latent variables.

When learning an HNB we can restrict our attention to the *parsimonious* HNB models; we need not consider models which encode a probability distribution that is also encoded by another model which has fewer parameters. Formally, an HNB model, $H = (B_S, \Theta_{B_S})$, with class variable C and attribute variables $\mathcal{A}$ is said to be parsimonious if there does not exist another HNB model, $H' = (B'_S, \Theta'_{B_S})$, with the same class and attribute variables s.t.:

i) H' has fewer parameters than H, i.e., $|\Theta_{B_S}| > |\Theta'_{B_S}|$.

ii) The probability distributions over the class and attribute variables are the same in the two models, i.e., $P(C, \mathcal{A}|B_S, \Theta_{B_S}) = P(C, \mathcal{A}|B'_S, \Theta'_{B_S})$.

In order to obtain an operational characterization of these models, Zhang et al. (2002) define the class of *regular* HNB models. An HNB model is said to be regular if for any latent variable L, with neighbours (parent and children) $X_1, X_2, \ldots X_n$, it holds that:

$$|\mathrm{sp}(L)| \leq \frac{\prod_{i=1}^{n} |\mathrm{sp}(X_i)|}{\max_{i=1,\ldots,n} |\mathrm{sp}(X_i)|},$$

and strict inequality holds when L has only two neighbours and at least one of them is a latent node.

Zhang et al. (2002) show that i) any parsimonious HNB model is regular, and ii) for a given set of class and attribute variables, the set of regular HNB model structures is finite. Observe that these two properties ensure that when searching for an HNB model we only need to consider regular HNB models and we need not deal with infinite search spaces.

As opposed to other frameworks, such as NB or TAN models, an HNB can model any correlation among the attribute variables by simply choosing the state-spaces of the latent variables large enough (although the encoding is not necessarily done in a cost-effective manner in terms of model complexity); note that the independence statements are not always represented explicitly in the graphical structure, but are sometimes only encoded in the conditional probability tables. On the other hand, the TAN model, for instance, is particular efficient for encoding such statements but may fail to represent certain types of dependence relations among the attribute variables. A TAN model is, e.g., not able to represent the statement "$C = 1$ if and only if exactly two out of the three attributes A_1, A_2 and A_3 are in state 1".

4 Learning HNB classifiers

4.1 The main algorithm

Our search algorithm is based on a greedy search over the space of all HNBs; we initiate the search with an HNB model, H_0, and learn a sequence $\{H_k\}$, $k = 1, 2 \ldots$ of HNB models. The search is conducted s.t. at each step we investigate the *search boundary* of the current model (denoted $\mathcal{B}(H_k)$), i.e., the set of models that can be reached from H_k in a single step. From this set of models the algorithm always selects a model with a higher score than the current one; if no such model can be found, then the current model is returned (see Algorithm 1).

Algorithm 1 (Greedy search)

1. Initiate model search with H_0;
2. **For** $k = 0, 1, \ldots$
   (a) Select $H' = \arg \max_{H \in \mathcal{B}(H_k)} \mathrm{Score}(H \,|\, \mathcal{D}_N)$;
   (b) **If** $\mathrm{Score}(H' \,|\, \mathcal{D}_N) > \mathrm{Score}(H_k \,|\, \mathcal{D}_N)$ **then**:
   $H_{k+1} \leftarrow H'$; $k \leftarrow k + 1$;
   else
   return H_k;

In order to make the above algorithm operational we need to specify the score function $\mathrm{Score}(\cdot \,|\, \mathcal{D}_N)$ as well as the search operator (which again defines the search boundary).

The score-function is defined s.t. a high value corresponds to what is thought to be a structure with good classification qualities (as measured by the average loss on unseen data), i.e., $\mathrm{Score}(\mathsf{H} \mid \mathcal{D}_\mathsf{N})$ measures the "goodness" of H. Note that the algorithm makes sure that $\mathrm{Score}(\mathsf{H}_{k+1} \mid \mathcal{D}_\mathsf{N}) > \mathrm{Score}(\mathsf{H}_k \mid \mathcal{D}_\mathsf{N})$ for $k = 0, 1, \ldots$ which ensures convergence as long as the score is finite for all models. In order to apply a score metric that is closely related to what the search algorithm tries to achieve, we use the wrapper approach by Kohavi and John (1997). That is, we use cross validation (over the training set $\mathcal{D}_\mathsf{N}$) to estimate an HNB's classification accuracy on unseen data; notice that the test-set (if defined) is *not* used when the score is calculated.

The search operator is defined s.t. the HNB structure is grown incrementally. More specifically, if $\mathcal{L}_k$ is the set of latent variables in model H_k, then the set of latent variables in H_{k+1}, is enlarged s.t. $\mathcal{L}_{k+1} = \mathcal{L}_k \cup \{\mathsf{L}\}$, where L is a new latent variable. We restrict ourself to only considering candidate latent variables which are parents of two variables X and Y where $\{\mathsf{X}, \mathsf{Y}\} \subseteq \mathrm{ch}(\mathsf{C})$ in H_k. Hence, we define H_{k+1} as the HNB which is produced from H_k by including a latent variable L s.t. $\mathrm{pa}(\mathsf{L}) = \{\mathsf{C}\}$ and $\mathrm{pa}(\mathsf{X}) = \mathrm{pa}(\mathsf{Y}) = \{\mathsf{L}\}$; H_{k+1} is otherwise identical to H_k. Thus, the search boundary $\mathcal{B}(\mathsf{H}_k)$ consists of all models where exactly one latent variable has been added to H_k; there is one model in $\mathcal{B}(\mathsf{H}_k)$ for each possible definition of the state-space of each possible new latent variable. Finally, as our starting point, H_0, we use the NB model structure; this implies that each H_k is a tree with a binary internal structure, i.e., any latent node $\mathsf{L}' \in \mathcal{L}_k$ has exactly two children but the class node C may have up to n children. It is obvious that any distribution is in principle reachable by the search algorithm but, as the score function is multi-modal over the search space, the search will in general only converge towards a local optimum.

4.2 Restricting the search boundary

Unfortunately, $\mathcal{B}(\mathsf{H}_k)$ is too large for the search algorithm to efficiently examine all models. To overcome this problem we shall instead focus the search by only selecting a subset of the models in $\mathcal{B}(\mathsf{H}_k)$, and these models are then used to represent the search boundary. The idea is to pinpoint a few promising candidates in the search boundary without examining all models available. Basically the algorithm proceeds in two steps by first deciding where to include a latent variable, and then defining the state-space of the new latent variable:[1]

1. Find a candidate latent variable.

2. Select the state-space of the latent variable.

Note that when using this two-step approach for identifying a latent variable, we cannot use scoring functions such as the wrapper approach, MDL, or MDL_p in the *first* step; this step does not select a completely specified HNB.

Before describing the two steps in detail, recall that the algorithm starts out with an NB model, and that the goal is to introduce latent variables to improve upon that structure, i.e., to avoid "double-counting" of information when the independence statements of the NB model are violated.

4.2.1 Step 1: Finding a candidate latent variable

To facilitate the goal of the algorithm, a latent variable L is proposed as the parent of $\{\mathsf{X}, \mathsf{Y}\} \subseteq \mathrm{ch}(\mathsf{C})$ if the data points towards $\mathsf{X} \not\perp\!\!\!\perp \mathsf{Y} \mid \mathsf{C}$. That is, we consider variables that are strongly correlated given the class variable as indicating a promising position for including a latent variable; from this perspective there is no reason to introduce a latent variable as a parent of X and Y if $\mathsf{X} \perp\!\!\!\perp \mathsf{Y} \mid \mathsf{C}$. Hence, the variables that have the highest correlation given the class variable may be regarded

[1]Ideally, a candidate latent variable should be selected directly (that is, defining location *and* state-space at the same time), but this is computationally prohibitive.

as the most promising candidate-pair. More specifically, we calculate the conditional mutual information given the class variable, $I(\cdot,\cdot\,|\,C)$, for all (unordered) pairs $\{X,Y\} \subseteq \mathrm{ch}(C)$. However, as $I(X,Y\,|\,C)$ is increasing in both $|\,\mathrm{sp}(X)\,|$ and $|\,\mathrm{sp}(Y)\,|$ we cannot simply pick the pair $\{X,Y\}$ that maximizes $I(X,Y\,|\,C)$; this strategy would unintentionally bias the search towards latent variables with children having large domains. Instead we utilize that:

$$2N \cdot I(X,Y\,|\,C) \xrightarrow{\mathcal{L}} \chi^2_{|\,\mathrm{sp}(C)\,|\left(|\,\mathrm{sp}(X)\,|-1\right)\left(|\,\mathrm{sp}(Y)\,|-1\right)},$$

where $\xrightarrow{\mathcal{L}}$ means convergence in distribution as $N \to \infty$, see e.g. (Whittaker 1990). Finally, we calculate

$$Q(X,Y\,|\,\mathcal{D}_N) = P\left(Z \leq 2N \cdot I(X,Y\,|\,C)\right), \tag{6}$$

where Z is χ^2 distributed with $|\,\mathrm{sp}(C)\,|\,(|\,\mathrm{sp}(X)\,|-1)\,(|\,\mathrm{sp}(Y)\,|-1)$ degrees of freedom. The pairs $\{X,Y\}$ are ordered according to these probabilities, s.t. the pair with the highest probability is picked out. By selecting the pairs of variables according to $Q(X,Y\,|\,\mathcal{D}_N)$, the correlations are normalized w.r.t. the size differences in the state-spaces.

Unfortunately, to greedily select a pair of highly correlated variables as the children of a new latent variable is not always the same as improving classification accuracy, as can be seen from the example below:[2]

Example 1 Consider a classifier with binary attributes $\mathcal{A} = \{A_1, A_2, A_3\}$ (all with uniform marginal distributions) and target concept $C = 1 \Leftrightarrow \{A_1 = 1 \wedge A_2 = 1\}$. Assume that A_1 and A_2 are marginally independent but that $P(A_2 = A_3) = 0.99$. It then follows that:

$$P\left(Q(A_2, A_3\,|\,\mathcal{D}_N) > Q(A_1, A_2\,|\,\mathcal{D}_N)\right) \to 1$$

as N grows large (the uncertainty is due to the random nature of $\mathcal{D}_N$). Hence, the heuristic will not pick out $\{A_1, A_2\}$ which is most beneficial w.r.t.˙ classification accuracy, but will propose to add a variable L' with children $\mathrm{ch}(L') = \{A_2, A_3\}$.

4.2.2 Step 2: Selecting the state-space

To find the cardinality of a latent variable L, we use an algorithm similar to the one by Elidan and Friedman (2001): Initially, the latent variable is defined s.t. $|\,\mathrm{sp}(L)\,| = \prod_{X \in \mathrm{ch}(L)} |\,\mathrm{sp}(X)\,|$, where each state of L corresponds to exactly one combination of the states of the children of L. Let the states of the latent variable be labelled $l_1, \ldots, l_t$. We then iteratively collapse two states l_i and l_j into a single state l^* as long as this is "beneficial". Ideally, we would measure this benefit using the wrapper approach, but as this is computationally expensive we shall instead use the MDL_p score to approximate the classification accuracy. Let $H' = (B'_S, \boldsymbol{\Theta}_{B'_S})$ be the HNB model obtained from a model $H = (B_S, \boldsymbol{\Theta}_{B_S})$ by collapsing states l_i and l_j. Then l_i and l_j should be collapsed if and only if $\Delta_L(l_i, l_j\,|\,\mathcal{D}_N) = \mathrm{MDL}_p(H\,|\,\mathcal{D}_N) - \mathrm{MDL}_p(H'\,|\,\mathcal{D}_N) > 0$. For each pair (l_i, l_j) of states we therefore compute:

$$\begin{aligned}
\Delta_L(l_i, l_j\,|\,\mathcal{D}_N) &= \mathrm{MDL}_p(H|\mathcal{D}_N) - \mathrm{MDL}_p(H'|\mathcal{D}_N) \\
&= \frac{\log(N)}{2}\left(|\,\boldsymbol{\Theta}_{B_S}\,| - \left|\,\boldsymbol{\Theta}_{B'_S}\right|\right) + \sum_{i=1}^{N}\left[\log\left(P_{H'}(c^{(i)}|a^{(i)})\right) - \log\left(P_H(c^{(i)}|a^{(i)})\right)\right].
\end{aligned}$$

[2]This issue is also discussed in Section 6.

For the second term we first note that:

$$\begin{aligned}\sum_{i=1}^{N}\left[\log\left(P_{H'}(c^{(i)}|a^{(i)})\right)-\log\left(P_{H}(c^{(i)}|a^{(i)})\right)\right] &= \sum_{i=1}^{N}\log\frac{P_{H'}(c^{(i)}|a^{(i)})}{P_{H}(c^{(i)}|a^{(i)})} \\ &= \sum_{D\in\mathcal{D}_N:f(D,l_i,l_j)}\log\frac{P_{H'}\left(c^{D}|a^{D}\right)}{P_{H}\left(c^{D}|a^{D}\right)},\end{aligned} \tag{7}$$

where $f(D, l_i, l_j)$ is true if case D includes either $\{L = l_i\}$ or $\{L = l_j\}$; cases which does not include these states cancel out. This is also referred to as *local decomposability* in (Elidan and Friedman 2001), i.e., the gain of collapsing two states l_i and l_j is local to those states and it does not depend on whether or not other states have been collapsed. In order to avoid considering all possible combinations of the attributes we approximate the difference in predictive MDL as the difference w.r.t. the relevant subtree. The relevant subtree is defined by C together with the subtree having L as root:[3]

$$\begin{aligned}&\sum_{D\in\mathcal{D}_N:f(D,l_i,l_j)}\log\frac{P_{H'}\left(c^{D}|a^{D}\right)}{P_{H}\left(c^{D}|a^{D}\right)} \\ &\approx \log\prod_{c\in\mathrm{sp}(C)}\left[\left(\frac{N(c,l_i)}{N(l_i)}\right)^{N(c,l_i)}\cdot\left(\frac{N(c,l_j)}{N(l_j)}\right)^{N(c,l_j)}\Big/\left(\frac{N(c,l_i)+N(c,l_j)}{N(l_i)+N(l_j)}\right)^{N(c,l_i)+N(c,l_j)}\right],\end{aligned} \tag{8}$$

where $N(c, s)$ and $N(s)$ are the sufficient statistics, e.g., $N(c, s) = \sum_{i=1}^{N}\gamma(C = c, L = s : \mathbf{D}_i)$; $\gamma(C = c, L = s : \mathbf{D}_i)$ takes on the value 1 if $(C = c, L = s)$ appears in case $\mathbf{D}_i$, and 0 otherwise; $N(s) = \sum_{c\in\mathrm{sp}(C)} N(c, s)$. Note that Equation 8 is in fact an equality if the relationship between C and $\mathrm{ch}(C)$ satisfy *independence of causal influence* (Heckerman and Breese 1994).

States are collapsed in a greedy manner, i.e., we find the pair of states with highest $\Delta_L(l_i, l_j \mid \mathcal{D}_N)$ and collapse those two states if $\Delta_L(l_i, l_j \mid \mathcal{D}_N) > 0$. This is repeated (making use of local decomposability) until no states can be collapsed, see also Algorithm 2.

Algorithm 2 (Determine state-space of L)

1. Initiate state-space s.t. $|\mathrm{sp}(L)| = \prod_{X\in\mathrm{ch}(L)}|\mathrm{sp}(X)|$;
   Label the states s.t. each state corresponds to a unique combination of $\mathrm{ch}(L)$;
2. **For each** $\{l_i, l_j\} \subseteq \mathrm{sp}(L)$ **do**:
   Calculate $\Delta_L(l_i, l_j \mid \mathcal{D}_N)$;
3. Select $\{l_i', l_j'\} \subseteq \mathrm{sp}(L)$ s.t. $\Delta_L(l_i', l_j' \mid \mathcal{D}_N)$ is maximized;
4. **If** $\Delta_L(l_i', l_j' \mid \mathcal{D}_N) > 0$ **then**:
   Collapse states l_i' and l_j'; **goto** 2;
5. **Return** state-space of L.

It should be noted that Elidan and Friedman (2001) initialize their search with one state in L for each combination of the variables in the *Markov blanket* of L, whereas we use the smaller set of variables defined by $\mathrm{ch}(L)$. This is done to facilitate a semantic interpretation of the latent variables (described below), and it does not exclude any regular HNB models.[4]

[3]The relevant subtree can also be seen as the part of the classifier structure that is directly affected by the potential collapse of the states l_i and l_j.

[4]Note that we do not consider regular HNB models with singly connected latent variables.

Example 2 (Example 1 cont'd) The state-space of L' with $\mathrm{ch}(L') = \{A_2, A_3\}$ is collapsed by Algorithm 2 after L' is introduced. For large N the penalty term in MDL_p ensures that the state-space will be collapsed to two states mirroring the states of A_2 because L' will not significantly change the predictive likelihood from what the model previously held (note that $P(C = c \mid A_2, A_3, \mathcal{D}_N) \approx P(C = c \mid A_2, \mathcal{D}_N)$). Hence, by introducing L' we get a more robust classifier, where the classification noise introduced by A_3 is removed. The latent variable L'' with children $\mathrm{ch}(L'') = \{L', A_1\}$ will be introduced in the next iteration of Algorithm 1, and the target concept can eventually be learned.

An important side-effect of Algorithm 2 is that we can give a semantic interpretation to the state-spaces of the latent variables: $L \in \mathcal{L}$ *aggregates* the information from its children which is relevant for classification. If, for example, L is the parent of two binary variables A_1 and A_2, then Algorithm 2 is initiated s.t. L's state-space is $\mathrm{sp}(L) = \{A_1 = 0 \wedge A_2 = 0, A_1 = 0 \wedge A_2 = 1, A_1 = 1 \wedge A_2 = 0, A_1 = 1 \wedge A_2 = 1\}$. When the algorithm collapses states, we can still maintain an explicit semantics over the state-space, e.g., if the first and second state is collapsed we obtain a new state defined as $(A_1 = 0 \wedge A_2 = 0) \vee (A_1 = 0 \wedge A_2 = 1)$, i.e., $A_1 = 0$. Having such an interpretation can be of great importance when the model is put into use: The semantics allows a decision maker to inspect the "rules" that form the basis of a given classification. Through this insight she can consider whether the classification of the system should be overruled or accepted.

Another important aspect of the semantic interpretation, is that it allows us to *infer* data for the latent variables due to the deterministic relations encoded in the model. This fact provides us with a fast calculation scheme, as we "observe" all the variables in $\mathcal{A}$ *and* $\mathcal{L}$. Therefore, it also follows that we can represent the HNB classifier using only the class variable and its children. Hence, the representation we will utilize is a Naïve Bayes structure where the "attributes" are represented by the variables which occur as children of the class variable in the HNB model. It is simple to realize that the number of free parameters required to represent this structure equals:

$$|\Theta_{B_S}| = (|\mathrm{sp}(C)| - 1) + |\mathrm{sp}(C)| \sum_{X \in \mathrm{ch}(C)} (|\mathrm{sp}(X)| - 1), \tag{9}$$

see also (Kočka and Zhang 2002). Hence, the difference in predictive MDL (used in Algorithm 2) can be approximated by:

$$\begin{aligned} \Delta_L(l_i, l_j) \approx\ & \log_2(N) \frac{|\mathrm{sp}(C)|}{2} \\ & - \sum_{c \in \mathrm{sp}(C)} N(c, l_i) \log_2 \left(\frac{N(c, l_i)}{N(c, l_i) + N(c, l_j)} \right) \\ & - \sum_{c \in \mathrm{sp}(C)} N(c, l_j) \log_2 \left(\frac{N(c, l_j)}{N(c, l_i) + N(c, l_j)} \right) \\ & + N(l_i) \log \left(\frac{N(l_i)}{N(l_i) + N(l_j)} \right) + N(l_j) \log \left(\frac{N(l_j)}{N(l_i) + N(l_j)} \right). \end{aligned} \tag{10}$$

Note again that the approximation is exact if the relationship between C and the children of C can be modelled using independence of causal influence.

4.2.3 The search boundary

By following the two-step procedure described above, the focusing algorithm produces a single candidate model $H' \in \mathcal{B}(H_k)$ to represent the search boundary. However, from our experiments we have found that picking out a single model to represent the search boundary is not an adequate representation of $\mathcal{B}(H_k)$. We can easily solve this drawback in at least two different ways:

i) Go through the candidate latent nodes one at a time (in order of decreasing $Q(\cdot,\cdot \mid \mathcal{D}_N)$), and accept the first candidate model $H'' \in \mathcal{B}(H_k)$ for which $\text{Score}(H'' \mid \mathcal{D}_N) > \text{Score}(H_k \mid \mathcal{D}_N)$ in Step 2b of Algorithm 1.

ii) Limit the number of candidates used to represent the boundary to $\kappa > 1$ models, and do a greedy search over these models.

The first approach can be seen as a *hill-climbing* search, where we use Equation 6 to guide the search in the right direction. Step 2a will in this case not be a maximization over $\mathcal{B}(H_k)$, but merely a search for a model which can be accepted in Step 2b. In Step 2a the algorithm may have to visit all models in the boundary $\mathcal{B}'(H_k) \subset \mathcal{B}(H_k)$ where $\mathcal{B}'(H_k)$ is defined s.t. each possible latent node is represented by exactly one state-space specification, i.e., a total of $O(n^2)$ models. On the other hand, the second approach will only examine κ models in Step 2a. It follows that alternative $i)$ has higher computational complexity; in fact we may have to inspect $O(n^3)$ candidates before the algorithm terminates (Step 2 may be repeated $n-1$ times), and since inspecting each candidate latent variable involves costly calculations it may be computationally expensive. For the results reported in Section 5 we have therefore used the second alternative: A fixed number of candidate models ($\kappa = 10$) are selected from the search boundary, and the search proceeds as in Algorithm 1. The computational complexity of this approach is detailed in Section 4.3.

An immediate approach for implementing this refined algorithm would be to: 1) pick out the κ node pairs that have the strongest correlation (according to Equation 6), 2) find the associated state-spaces, and 3) select the model with the highest score in Step 2a. However, to increase the robustness of the algorithm, we do it slightly differently: Initially, we randomly partition the training data $\mathcal{D}_N$ in κ partly overlapping subsets, each containing $(\kappa-1)/\kappa$ of the training data, and then each of these subsets are used to approximate the *best* model in the search boundary; this results in a list of up to κ different candidate models. We let these models represent $\mathcal{B}(H_k)$, and continue as if this was the whole boundary: If the best model amongst them (the one with the highest accuracy estimated by cross validation over the training data) is better than the current model candidate, we select that one and start all over again. If the best model is inferior to the current model, the search algorithm terminates, and the current model is returned (see Algorithm 3).

Algorithm 3 (Find HNB classifier)

1. Initiate model search with H_0;
2. Partition the training-set into κ partly overlapping subsets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(\kappa)}$;
3. **For** $k = 0, 1, \ldots, n-1$
   (a) **For** $i = 1, \ldots, \kappa$
      i. Let $\{X^{(i)}, Y^{(i)}\} = \arg\max_{\{X,Y\} \subseteq \text{ch}(C)} Q\left(X, Y \mid \mathcal{D}^{(i)}\right)$ (i.e., $\{X^{(i)}, Y^{(i)}\} \subseteq \text{ch}(C)$ in H_k), and define the latent variable $L^{(i)}$ with children $\text{ch}\left(L^{(i)}\right) = \{X^{(i)}, Y^{(i)}\}$.
      ii. Collapse the state-space of $L^{(i)}$ (Algorithm 2 with $\mathcal{D}^{(i)}$ used in place of $\mathcal{D}_N$);
      iii. Define $H^{(i)}$ by introducing $L^{(i)}$ into H_k;
   (b) $H' = \arg\max_{i=1,\ldots,\kappa} \text{Score}\left(H^{(i)} \mid \mathcal{D}_N\right)$;
   (c) **If** $\text{Score}(H' \mid \mathcal{D}_N) > \text{Score}(H_k \mid \mathcal{D}_N)$ **then**:
      $H_{k+1} \leftarrow H'$; $k \leftarrow k+1$;
      else
      return H_k;
4. **Return** H_n;

4.3 Complexity analysis

When analyzing the complexity of the algorithm we can divide the description into three steps:

1) Find a candidate latent variable.

2) Find the state-space of a candidate latent variable, and check if it is useful.

3) Iterate until no more candidate latent variables are accepted.

Part 1
Proposing a candidate latent variable corresponds to finding the pair (X, Y) of variables having the highest conditional mutual information. There are at most $(n^2 - n)/2$ such pairs, where n is the number of attribute variables. Calculating the conditional mutual information for a pair of variables can be done in time $O(N)$ (N being the number of cases in the database) hence, calculating the conditional mutual information for each pair of variables can be done in time $O(N \cdot n^2)$. Finally, the list is sorted (to accommodate future iterations), and the resulting time complexity is $O(n^2 \cdot (N + \log(n)))$.

Part 2
When determining the cardinality of a latent variable, L, we consider the gain of collapsing two states as compared to the current model; the gain is measured as the difference in predictive MDL. The time complexity of calculating the gain of collapsing two states is simply $O(N)$, see Equation 10. Due to local decomposability, the gain of collapsing two states has no effect on collapsing two other states, and there are therefore at most $(|\,\mathrm{sp}\,(L)\,|^2 - |\,\mathrm{sp}\,(L)\,|)/2$ possible combinations, i.e., $O(|\,\mathrm{sp}\,(L)\,|^2 \cdot N)$. When two states are collapsed, $\Delta_L(\cdot, \cdot)$ must be calculated for $|\,\mathrm{sp}\,(L)\,| - 1$ new state combinations, next time $|\,\mathrm{sp}\,(L)\,| - 2$ state combinations are evaluated, and so on; the collapsing is performed at most $|\,\mathrm{sp}\,(L)\,| - 1$ times. The time complexity of finding the state-space of a candidate latent variable is therefore $O\left(N \cdot |\,\mathrm{sp}\,(L)\,|^2 + N \cdot |\,\mathrm{sp}\,(L)\,|\,(|\,\mathrm{sp}\,(L)\,| - 1)/2\right) = O(|\,\mathrm{sp}\,(L)\,|^2 \cdot N)$.

Having found the cardinality of a candidate variable, say L, we test whether it should be included in the model using the wrapper approach. From the rule-based propagation method it is easy to see that the time complexity of this task is $O(n \cdot N)$. Thus, the time complexity of Part 2 is $O((n + |\,\mathrm{sp}\,(L)\,|^2) \cdot N)$.

Part 3
Each time a latent variable is introduced we would in principle need to perform the above steps again, and the time complexity would therefore be $n - 1$ times the time complexities above. However, as described below some of the previous calculations can be reused.

First of all, as $Q(X, Y|\mathcal{D})$ is a local measure we only need to calculate $Q(L, Z|\mathcal{D})$, $Z \in \mathrm{ch}\,(C)$, where L is the latent variable introduced in the previous iteration. Moreover, since we need to calculate $Q(L, \cdot|\mathcal{D})$ at most $n - 2$ times, the time complexity will be $O(n \cdot N)$, and, as the pairs (X, Y) are still sorted according to $Q(X, Y|\mathcal{D})$, we only need to sort $n - 2$ pairs, i.e., after having included a latent variable the re-initialization of step 1 has complexity $O(n \cdot N + (n - 1) \cdot \log(n - 1)) = O(n \cdot (N + \log(n)))$.

Moreover, after having introduced a latent variable L with children X and Y, we cannot create another latent variable having either X or Y as a child (due to the structure of the HNB model). Thus, after having included a latent variable the cardinality of the resulting set of candidate pairs is reduced by $n - 1$. This implies that we will perform at most $n - 2$ re-initializations, thereby giving the overall time complexity $O(n^2 \cdot N + n \cdot (n \cdot (N + \log(n)) + (|\,\mathrm{sp}\,(L)\,|^2 \cdot N))) = O(n^2 \cdot (\log(n) + |\,\mathrm{sp}\,(L)\,|^2 \cdot N))$.

5 Empirical results

In this section we will investigate the merits of the proposed learning algorithm by using it to learn classifiers for a number of different domains. All data-sets are taken from the Irvine Machine Learning Repository (Blake and Merz 1998), see Table 1 for a summary of the 22 datasets used in this empirical study.

			#Inst					#Inst	
Database	#Att	#Cls	Train	Test	Database	#Att	#Cls	Train	Test
postop	8	3	90	CV(5)	cleve	13	2	296	CV(5)
iris	4	3	150	CV(5)	wine	13	3	178	CV(5)
monks-1	6	2	124	432	thyroid	5	3	215	CV(5)
monks-2	6	2	124	432	ecoli	7	8	336	CV(5)
monks-3	6	2	124	432	breast	10	2	683	CV(5)
glass	9	7	214	CV(5)	vote	16	2	435	CV(5)
glass2	9	2	163	CV(5)	crx	15	2	653	CV(5)
diabetes	8	2	768	CV(5)	australian	14	2	690	CV(5)
heart	13	2	270	CV(5)	chess	36	2	2130	1066
hepatitis	19	2	155	CV(5)	vehicle	18	4	846	CV(5)
pima	8	2	768	CV(5)	soybean-large	35	19	562	CV(5)

Table 1: A summary of the 22 databases used in the experiments: #Att indicates the number of attributes; #Cls is the number of classes; #Inst is the number of instances (given separately for training and test sets). CV(5) denotes 5-fold cross validation. Further details regarding the datasets can be found at the UCI Machine Learning Repository.

We have compared the results of the HNB classifier to those of the Naïve Bayes model (Duda and Hart 1973), the TAN model (Friedman et al. 1997), C5.0 (Quinlan 1998), and a standard implementation of neural networks with one hidden layer trained by back-propagation.[5] As some of the learning algorithms require discrete variables, the attributes were discretized using the entropy-based method of (Fayyad and Irani 1993). In addition, instances containing missing attribute-values were removed; all pre-processing was performed using MLC++ (Kohavi et al. 1994).

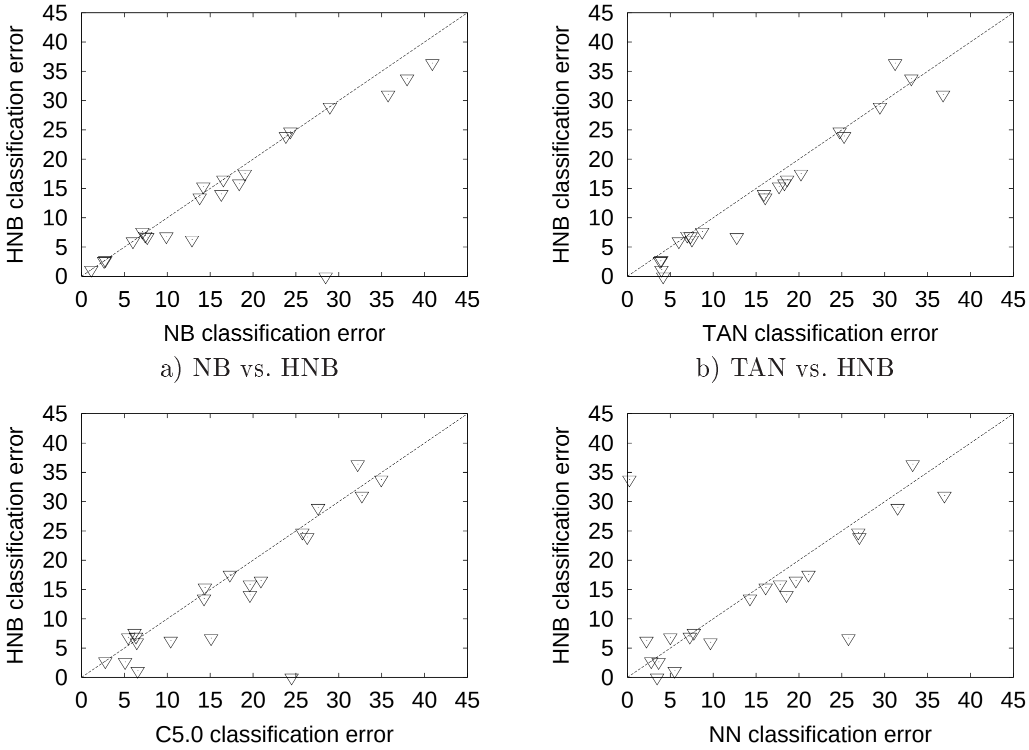
The accuracy-results are given in Table 2. For each dataset we have estimated the accuracy of each classifier (in percentage of instances which are correctly classified), and give a standard deviation of this estimate. The standard deviations are the theoretical values calculated according to (Kohavi 1995), and are not necessarily the same as the empirical standard deviations observed during cross validation. For comparison of the algorithms we made sure that the same cross validation folds were used for all the different learning methods. The best result for each dataset is given in boldface. We note that the HNB classifier achieves the best result for 10 of the 22 datasets, comes top-two for all but 5 datasets, and also has the best performance averaged over all datasets.

To quantify the difference between the HNB classifier and the other classifiers we advocate the method of (Kohavi 1995); Kohavi (1995) argues that the true merit of a classifier cannot be found by calculating the accuracy on a finite test-set. Instead we define α as the true accuracy of a classifier (only to be found if the target concept of the domain is known or fully described by an infinite test set), and we use $\hat{\alpha}$ to denote the estimate of α based on a test set of size N. Kohavi (1995) argues that $\hat{\alpha}$ is approximately Gaussian distributed with expectation α and variance $\alpha \cdot (1-\alpha)/N$ for large N. In our setting we have several datasets (indexed by $i = 1, \ldots, t$; t is

[5]We used Clementine (SPSS Inc. 2002) to generate the C5.0 and neural network models. We have not compared our system to that of (Zhang et al. 2002) because of the high computational complexity of Zhang et al.'s algorithm. However, the numerical results reported by Zhang et al. (2002) point towards our model offering significantly better classification accuracy.

Database	NB	TAN	C5.0	NN	HNB
postop	64.25+/-5.0	63.20+/-5.1	67.31+/-4.9	63.04+/-5.1	**68.95+/-4.9**
iris	**94.00+/-2.0**	**94.00+/-2.0**	93.55+/-2.0	90.32+/-2.4	**94.00+/-2.0**
monks-1	71.53+/-2.2	95.83+/-1.0	75.50+/-2.1	96.54+/-0.9	**100.0+/-0.1**
monks-2	62.04+/-2.3	66.90+/-2.3	65.05+/-2.3	**99.77+/-0.3**	66.20+/-2.0
monks-3	**97.22+/-0.8**	96.06+/-0.9	**97.22+/-0.8**	**97.22+/-0.8**	**97.22+/-0.8**
glass	71.04+/-3.1	70.56+/-3.1	**72.42+/-3.1**	68.50+/-3.2	71.04+/-3.1
glass2	81.61+/-3.0	81.69+/-3.0	80.37+/-3.1	82.21+/-3.0	**84.11+/-3.1**
diabetes	**75.65+/-1.5**	75.25+/-1.6	74.25+/-1.6	73.08+/-1.6	75.25+/-1.5
heart	83.70+/-2.2	84.07+/-2.2	80.36+/-2.4	81.45+/-2.4	**85.93+/-2.3**
hepatitis	92.34+/-2.1	87.25+/-2.7	84.89+/-2.9	74.23+/-3.5	**93.29+/-2.1**
pima	**76.17+/-1.5**	74.74+/-1.6	73.68+/-1.6	72.96+/-1.6	76.04+/-1.5
cleve	**83.46+/-2.1**	81.38+/-2.2	79.08+/-2.4	80.36+/-2.3	83.45+/-2.2
wine	**98.86+/-0.8**	96.03+/-1.5	93.45+/-1.9	94.49+/-1.7	**98.86+/-0.8**
thyroid	92.56+/-1.8	93.02+/-1.7	**93.64+/-1.7**	92.73+/-1.8	93.02+/-1.7
ecoli	80.95+/-2.1	79.76+/-2.2	**82.70+/-2.1**	78.89+/-2.2	82.44+/-2.1
breast	**97.36+/-0.6**	96.19+/-0.7	94.92+/-0.8	96.36+/-0.7	**97.36+/-0.6**
vote	90.11+/-1.4	92.64+/-1.3	94.55+/-1.1	**95.00+/-1.1**	93.15+/-1.3
crx	86.22+/-1.3	83.93+/-1.4	85.71+/-1.4	85.71+/-1.4	**86.51+/-1.3**
australian	**85.80+/-1.3**	82.32+/-1.5	85.61+/-1.3	83.88+/-1.4	84.64+/-1.4
chess	87.12+/-1.0	92.48+/-0.8	89.60+/-0.9	**97.78+/-0.5**	93.71+/-0.7
vehicle	59.09+/-1.7	**68.79+/-1.6**	67.80+/-1.6	66.74+/-1.6	63.59+/-1.7
soybean-large	92.90+/-1.0	91.28+/-1.1	**93.82+/-1.0**	92.25+/-1.1	92.36+/-1.1
Average	82.91	83.97	82.98	84.71	85.52

Table 2: Calculated accuracy for the 22 datasets used in the experiments. The results are given together with their theoretical standard deviation.



a) NB vs. HNB

b) TAN vs. HNB

c) C5.0 vs. HNB

d) NN vs. HNB

Figure 2: Scatter plot of classification error for HNB and a selection of other classification systems. In each plot, a point represents a dataset. The HNB's classification error is given on the x-axis, whereas the other system's error is given on the y-axis. Hence, data points below the diagonal corresponds to datasets where the HNB is superior, whereas points above the diagonal are datasets where the HNB classifier is inferior to the other system.

the number of datasets, i.e., $t = 22$ in this study) and several classifier algorithms (indexed by j), and with this notation Kohavi's approximation can be written as $\hat{\alpha}_{ij} \sim \mathcal{N}(\alpha_{ij}, \alpha_{ij} \cdot (1 - \alpha_{ij})/N_i)$. To simplify, we assume $\hat{\alpha}_{ij} \perp\!\!\!\perp \hat{\alpha}_{ik}$ for $j \neq k$ and $\hat{\alpha}_{ij} \perp\!\!\!\perp \hat{\alpha}_{\ell j}$ for $i \neq \ell$. Finally, we use the estimated standard deviation s_{ij} (given in Table 2) as if it was known. It follows that under the hypothesis that classifiers j and k are equally capable ($\alpha_{ij} = \alpha_{ik}$, $i = 1, \ldots, t$) then:

$$\Lambda_i(j,k) = \hat{\alpha}_{ij} - \hat{\alpha}_{ik} \sim \mathcal{N}(0, s_{ij}^2 + s_{ik}^2) \ , \quad \Lambda(j,k) = \sum_{i=1}^{t} \frac{\Lambda_i}{t} \sim \mathcal{N}\left(0, \sum_{i=1}^{t} \frac{s_{ij}^2 + s_{ik}^2}{t^2}\right) \ .$$

This enables us to test the hypothesis that the HNB classifier is not better than the other classifiers; more precisely we test the hypothesis H_0: $\Lambda(\cdot,\cdot) \leq 0$ against H_1: $\Lambda(\cdot,\cdot) > 0$, where the classifiers are labelled s.t. higher average accuracy for the HNB classifier coincides with a positive value of $\Lambda(\cdot,\cdot)$. With this setup H_0 is rejected at level $p = 5 \cdot 10^{-12}$ (NB), $p = 6 \cdot 10^{-6}$ (TAN), $p = 6 \cdot 10^{-11}$ (C5.0) and $p = .02$ (NN).

Finally, we note that in some of the domains the HNB models come up with an interesting latent structure. We are not experts to tell whether these structures are in fact meaningful, but some of them are at least worth attention. For example, in the heart model the HNB aggregates information about "Chest pain" and "Training induced angina". The probability of a heart disease increases slightly when chest pain is of a certain type; this probability can then again be increased dramatically if the instance also contains information about a training induced angina. Training induced angina has no effect in the model if chest pain is not of this particular type. Note that the classifier in this example uses the latent variable to encode *context specific independence* (Boutilier et al. 1996).

6 Discussion

6.1 Parameter learning

The parameters in the model are estimated by their maximum likelihood values. This may not be optimal for classification, and recent research has shown some improvement in classification accuracy when the parameters are chosen otherwise (Wettig et al. 2002). However, to support the interpretation of the empirical results in Section 5 we have deliberately not taken the opportunity of improving the classification accuracy further in this way. Optimization of the model is left for future work.

6.2 Finding candidate latent variables

As described by Example 1 and Example 2 the search for candidate latent variables may introduce a latent variable for a pair of variables which are marginally dependent, but where only one of the variables is actually dependent on the class variable C; as also shown is the examples, this does not jeopardize classification accuracy (actually it can be seen as a form of feature selection). Similarly, if several attributes are marginally dependent but independent of the class variable, the algorithm performs some redundant computations: For each such pair of attributes we include a latent variable, but as these attributes are independent of the class variable, all states of such a latent variable are collapsed and the effect of the attributes on the classification result is removed.

Obviously both of the above mentioned problems can be overcome by simply performing a feature selection before initializing the learning algorithm. However, another approach would be to apply a correlation measure which directly considers the probability distribution over the class variable conditioned on the two variables X and Y in question. That is, the difference between the probability distribution $P(C|X,Y)$ and the probability distribution $P'(C|X,Y)$, where the latter is encoded by the model where $X \perp\!\!\!\perp Y|C$. This distance can be described using the well-known Kullback-Leibler (KL) divergence (Kullback and Leibler 1951) averaged over the possible states of X and Y:

$$\mathbb{E}(KL(P;P')|X,Y) = \sum_{x,y} P(x,y) \sum_{c} P(c|x,y) \log\left(\frac{P(c|x,y)}{P'(c|x,y)}\right).$$

In the context of classification, this distance measure can also be given another interpretation by observing that:

$$\begin{aligned}
\mathbb{E}(KL(P;P')|X,Y) &= \sum_{c,x,y} P(c,x,y) \log\left(\frac{P(c,x,y)}{P(x,y)} \cdot \frac{1}{P'(c|x,y)}\right) \\
&= \sum_{c,x,y} P(c,x,y) \log\left(\frac{P(c,x,y)}{P(x,y)} \cdot \frac{\sum_c (P(x|c)P(y|c)P(c))}{P(x|c)P(y|c)P(c)}\right) \\
&= \sum_{x,y,c} P(x,y,c) \log\left(\frac{P(x,y|c)}{P(x|c)P(y|c)}\right) - \sum_{x,y} P(x,y) \log\left(\frac{P(x,y)}{\sum_c P(x|c)P(y|c)P(c)}\right) \\
&= I(X,Y|C) - KL(P(X,Y), P'(X,Y)).
\end{aligned} \tag{12}$$

Thus, the expected KL-divergence can be interpreted as the difference in conditional mutual information between X and Y conditioned on C, and the KL-divergence between $P(X,Y)$ in the unconstrained model and the model where $X \perp\!\!\!\perp Y | C$. In particular, if X and Y are marginally dependent but independent of the class variable C, we would have $\mathbb{E}(KL(P; P')|X, Y) = 0$ whereas $I(X, Y|C) > 0$ would have suggested that a latent variable should be introduced. Thus, this distance measure also takes into account that variables may be marginally dependent but independent of the class variable.

6.3 Inference and model structure

The algorithm for collapsing the state-space of a latent variable is the source of the semantics for these nodes, and in turn the reason why we can represent the HNB as a Naïve Bayes model with aggregations in place of the attributes. This compact representation requires a "deterministic inference engine" to calculate $P(C \mid \mathbf{a})$, because the aggregations defined by the semantics of the latent variables can in general not be encoded by the conditional probability tables for the variables. Assume, for instance, that we have three binary variables L, X, Y, $\mathrm{ch}(L) = \{X, Y\}$, and "$L = 1$ if and only if $X = Y$". This relationship cannot be encoded in the model $X \leftarrow L \rightarrow Y$, and to infer the state of the latent variable L from X and Y we would therefore need to design a special inference algorithm which explicitly uses the semantics of L. To alleviate this potential drawback we can simply re-define the network-structure: Introduce a new latent variable L', and change the network structure s.t. $\mathrm{ch}(L) = \mathrm{pa}(X) = \mathrm{pa}(Y) = \{L'\}$; L' is equipped with at most one state for each possible combination of its children's states. This enlarged structure is capable of encoding any relation between $\{X, Y\}$ and L using the conditional probability tables only. Hence, the enlarged structure can be handled by any standard BN propagation algorithm and, since the structure is still an HNB, the inference can be performed extremely fast.

7 Concluding remarks

In this paper we have used Hierarchical Naïve Bayes models for classification, and through experiments we have shown that the HNB classifiers offer results that are significantly better than those of other commonly used classification methods. Moreover, a number of existing tools may be able to improve the classification accuracy even further. These include feature selection (Kohavi and John 1997), smoothing (significant improvements reported by (Friedman et al. 1997) for some model classes), and supervised learning of the probability parameters (Wettig et al. 2002). We leave the investigation of these sources of potential improvements for future work. Finally, the proposed learning algorithm also provides an explicit semantics for the latent structure of a model. This allows a decision maker to easily deduce the rules which govern the classification of some instance hence, the semantics may also increase the user's confidence in the model.

Acknowledgements

We have benefited from interesting discussions with the members of the Decision Support Systems group at Aalborg University, in particular Tomás Kočka, Nevin L. Zhang, and Jiří Vomlel. We would like to thank `Hugin Expert` (www.hugin.com) for giving us access to *Hugin Decision Engine* which forms the basis for our implementation. The first author was supported by a grant from the Research Council of Norway.

References

Blake, C. and C. Merz (1998). UCI repository of machine learning databases. URL: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Boutilier, C., N. Friedman, M. Goldszmidt, and D. Koller (1996). Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, Portland, OR., pp. 115–123.

Chow, C. K. and C. Liu (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory 14*, 462–467.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Sciences. New York: Springer Verlag.

Domingos, P. and M. Pazzani (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29*(2–3), 103–130.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Elidan, G. and N. Friedman (2001). Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 144–151. Morgan Kaufmann Publishers.

Fayyad, U. M. and K. B. Irani (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, San Mateo, CA., pp. 1022–1027. Morgan Kaufmann Publishers.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse of dimensionality. *Data Mining and Knowledge Discovery 1*(1), 55–77.

Friedman, N., D. Geiger, and M. Goldszmidt (1997). Bayesian network classifiers. *Machine Learning 29*(2–3), 131–163.

Greiner, R., A. J. Grove, and D. Schuurmans (1997). Learning Bayesian nets that perform well. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 198–207. Morgan Kaufmann Publishers.

Heckerman, D. and J. S. Breese (1994). A new look at causal independence. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 286–292. Morgan Kaufmann Publishers.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. New York: Springer Verlag.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, San Mateo, CA., pp. 1137–1143. Morgan Kaufmann Publishers.

Kohavi, R., G. John, R. Long, D. Manley, and K. Pfleger (1994). MLC++: A machine learning library in C++. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, pp. 740–743. IEEE Computer Society Press.

Kohavi, R. and G. H. John (1997). Wrappers for feature subset selection. *Artificial Intelligence 97*(1–2), 273–324.

Kononenko, I. (1991). Semi-naive Bayesian classifier. In *Proceedings of Sixth European Working Session on Learning*, Berlin. Springer Verlag.

Kočka, T. and N. L. Zhang (2002). Dimension correction for hierarchical latent class models. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA., pp. 267–274. Morgan Kaufmann Publishers.

Kullback, S. and R. A. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics 22*, 79–86.

Lam, W. and F. Bacchus (1994). Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence 10*(4), 269–293.

Mitchell, T. M. (1997). *Machine Learning*. Boston, MA.: McGraw Hill.

Pazzani, M. (1995). Searching for dependencies in Bayesian classifiers. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics.*

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA.: Morgan Kaufmann Publishers.

Quinlan, R. (1998). C5.0: An informal tutorial. http://www.rulequest.com/see5-unix.html.

Rissanen, J. (1978). Modelling by shortest data description. *Automatica 14*, 465–471.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics 6*, 461–464.

SPSS Inc. (2002). Clementine v6.5. http://www.spss.com/spssbi/clementine/index.htm.

Wettig, H., P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri (2002). On supervised learning of Bayesian network parameters. HIIT Technical Report 2002-1, Helsinki Institute for Information Technology.

Whittaker, J. (1990). *Graphical models in applied multivariate statistics.* Chichester: John Wiley & Sons.

Zhang, N. (2002). Hierarchical latent class models for cluster analysis. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, Menlo Park, CA., pp. 230–237. AAAI Press.

Zhang, N., T. D. Nielsen, and F. V. Jensen (2002). Latent variable discovery in classification models. Available from the first author upon request.